Pods e Serviços

Pedro Horchulhack

Agenda

1. Pods
2. Serviços
3. Atividade

Pods

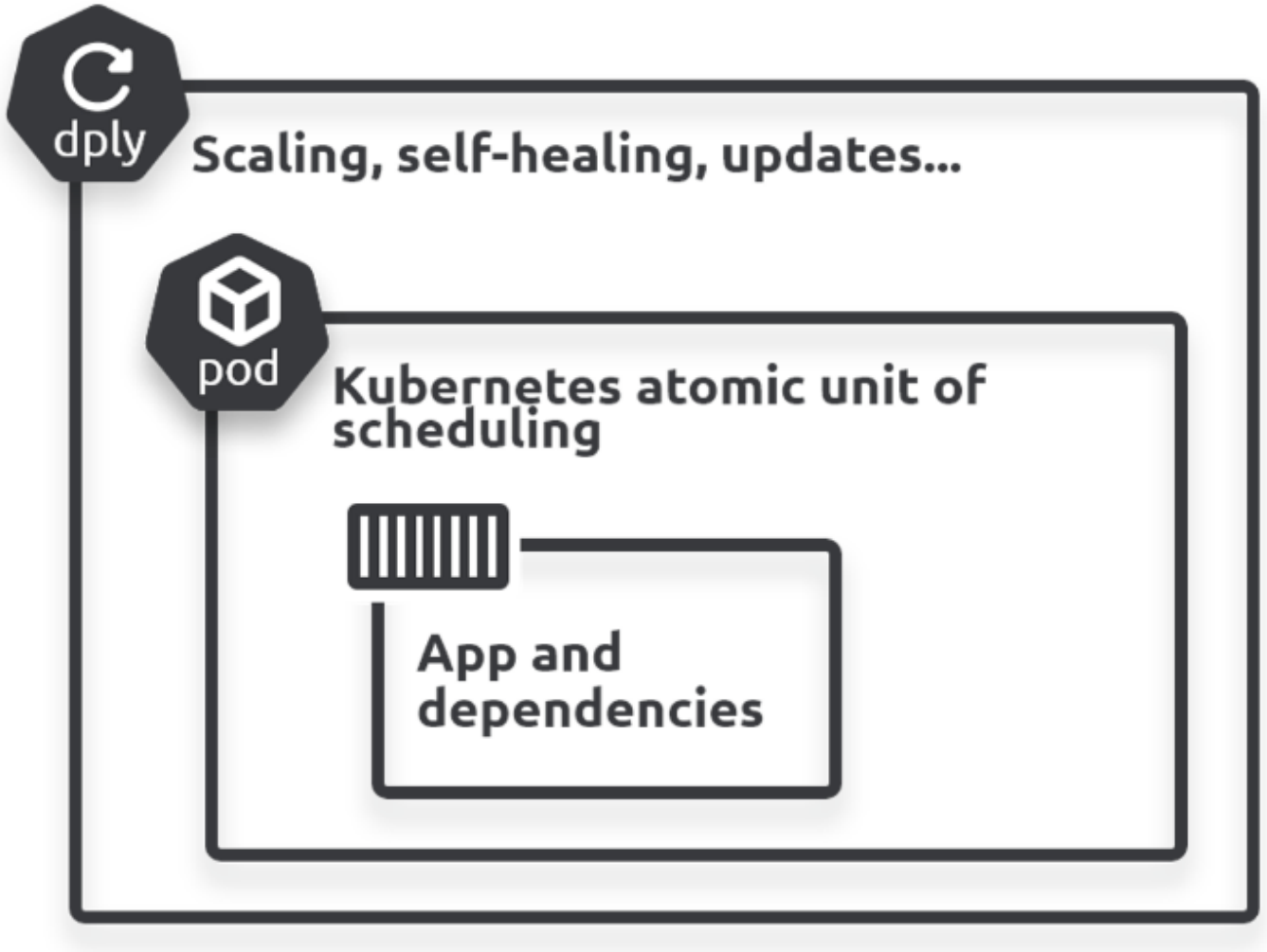
Pods

- Pod é a menor unidade de controle dentro do Kubernetes (Pod com a primeira letra maiúscula)
  - Atua como um envelopador de um contêiner para deixá-lo apto a executar no K8S
  - É a menor unidade escalonável no sistema

- Envelopa contêineres para serem executados no Kubernetes

- O modelo mais simples é um contêiner por Pod
  - Podemos instanciar mais de um contêiner em um Pod
  - Contêineres (2 ou mais) no mesmo Pod executam no mesmo nó

Pods

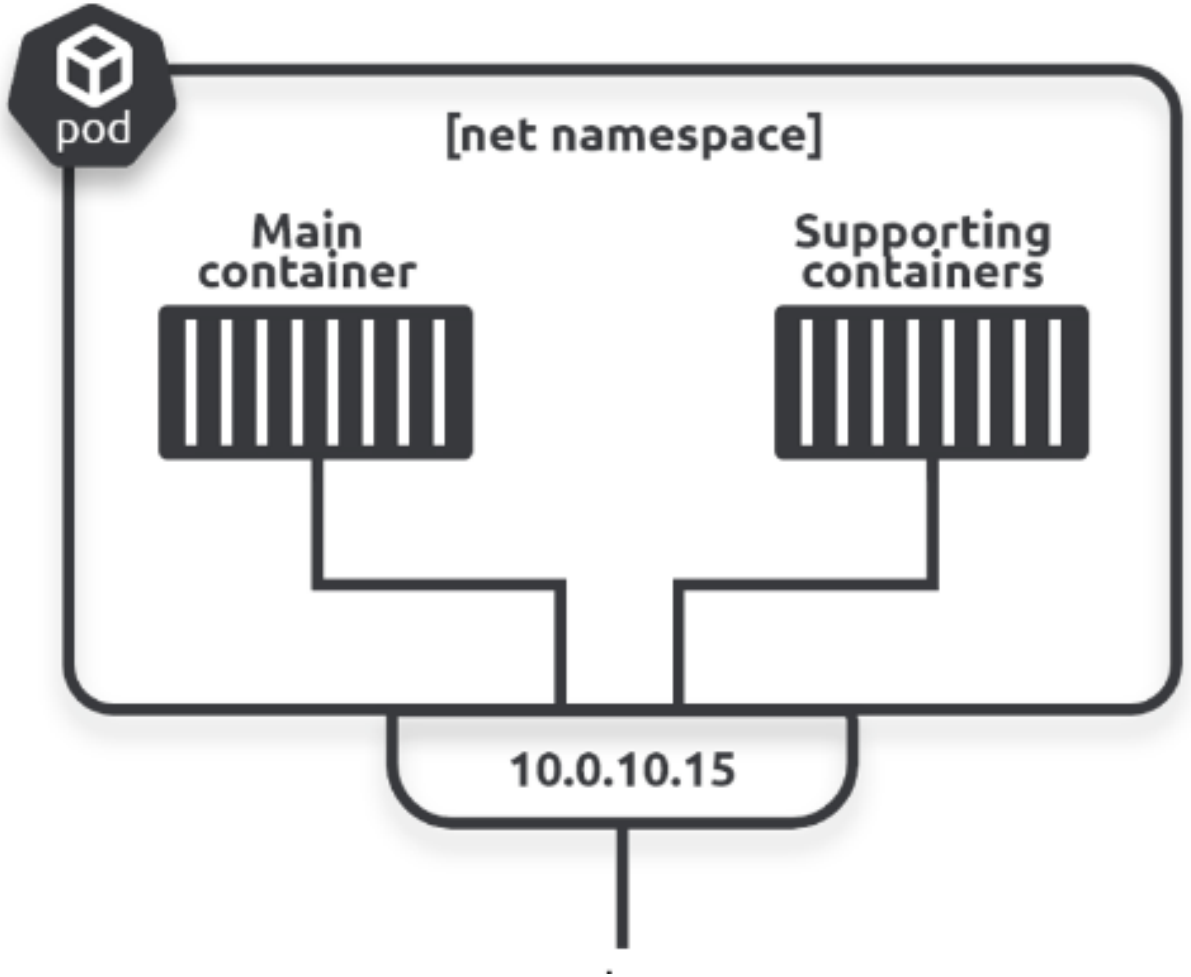


Fonte: POULTON, Nigel; JOGLEKAR, Pushkar. **The Kubernetes book**. Version 4, March 2019. United Kingdom: Nigel Poulton, 2019.

Pods

- Lembrando que Pods não executam aplicações, mas os contêineres dele

- Se tivermos mais de um contêiner executando no mesmo Pod eles compartilharão
  - Memória
  - Volumes
  - Sistemas de arquivos
  - Pilha de rede



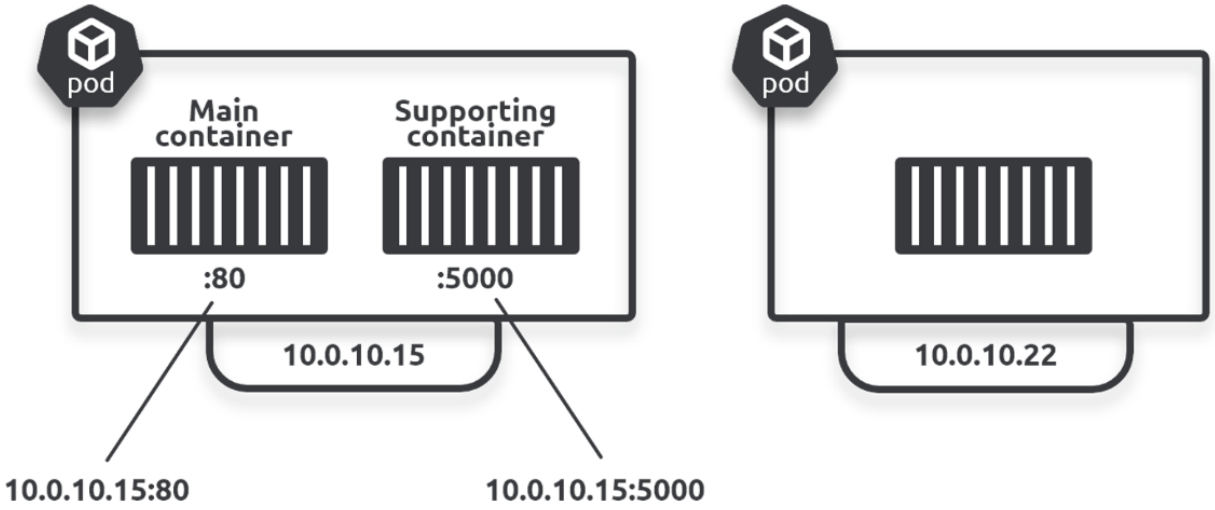
Fonte: POULTON, Nigel; JOGLEKAR, Pushkar. **The Kubernetes book**. Version 4, March 2019. United Kingdom: Nigel Poulton, 2019.

Pods

- Pods possuem um *namespace* de rede próprio

- Pod com um contêiner
  - O IP tem acesso completo ao IP e as portas

- Pod com dois ou mais contêineres
  - Compartilham o mesmo IP, tabela de roteamento e intervalo de portas



Fonte: POULTON, Nigel; JOGLEKAR, Pushkar. **The Kubernetes book**. Version 4, March 2019. United Kingdom: Nigel Poulton, 2019.

Pods

- Pods possuem um ciclo de vida simples:
  - São criados (created/waiting)
  - Vivem (running)
  - Morrem (terminated)

- Se eles morrerem “do nada”, o Kubernetes irá criar um Pod novo, com um novo ID e IP
  - Isso implica no desenvolvimento das aplicações
  - O objetivo agora é desenvolver aplicações fracamente acopladas internamente

- Também são imutáveis. Não podemos alterá-lo uma vez executando.
  - Caso queiramos atualizá-lo, precisamos removê-lo e criá-lo novamente

Pods

- Podemos instanciar Pods de duas maneiras
  - Diretamente através de um manifesto (Estático)
  - Indiretamente através de um controlador (Dinâmico)

- Pods estáticos não possuem tantas vantagens
  - Somente um único agente irá monitorá-lo (Não permite inicializar em outro nó)

- Pods dinâmicos são controlados e sua substituição é mais eficiente
  - Pode ocorrer em outros nós

- Você geralmente utilizará controladores para instanciar Pods

Pods – Instanciação

A instanciação de um Pod ocorre seguindo:

1. A sua definição em um manifesto YAML
2. Enviar o YAML a API
3. API autentica e autoriza a requisição
4. A configuração é validada
5. O escalonador instancia o Pod em um nó saudável com recursos suficientes e disponíveis
6. O Kubelet local monitora ele

```
apiVersion: v1
kind: Pod
metadata:
  name: aula-pods
  labels:
    regiao: sul
    versao: beta
spec:
  containers:
    - name: ubuntu
      image: ubuntu:22.04
      command: ['tail', '-f', '/dev/null']
  restartPolicy: OnFailure
```

Pods – Instanciação

Vamos executar o manifesto ao lado

1. Copie em um arquivo e renomei-o para `pod.yaml`
2. Crie o pod

   ```
   kubectl apply -f pod.yaml
   ```
3. Para verificar se está executando

   ```
   kubectl get pods –o wide
   ```
4. Ver descrição

   ```
   kubectl describe pod aula-pods
   ```
5. Para remover o Pod

   ```
   kubectl delete -f pod.yaml
   ```

```
apiVersion: v1
kind: Pod
metadata:
  name: aula-pods
  labels:
    regiao: sul
    versao: beta
spec:
  containers:
    - name: ubuntu
      image: ubuntu:22.04
      command: ['tail', '-f', '/dev/null']
  restartPolicy: OnFailure
```

Pods

- Essa estrutura nos fornece alguns complementos aos contêineres
  - Labels e anotações
  - Políticas de restart
  - Probes (inicialização, prontidão, vivacidade, …)
  - Regras de afinidade e anti-afinidade
  - Controle de terminação
  - Políticas de segurança
  - Solicitações e limites de recursos

- O comando **`kubectl explain pods --recursive`** nos dá uma visão geral das configurações
  - Também podemos escolher alguma configuração para o kubectl explicar

Pods

```
ubuntu@arm:~$ kubectl explain pods --recursive | head -n 50
KIND:       Pod
VERSION:    v1

DESCRIPTION:
    Pod is a collection of containers that can run on a host. This resource is
    created by clients and scheduled onto hosts.

FIELDS:
  apiVersion    <string>
  kind  <string>
  metadata      <ObjectMeta>
    annotations <map[string]string>
    creationTimestamp   <string>
    deletionGracePeriodSeconds  <integer>
    deletionTimestamp   <string>
    finalizers  <[]string>
    generateName        <string>
    generation  <integer>
    labels      <map[string]string>
    managedFields       <[]ManagedFieldsEntry>
      apiVersion        <string>
      fieldsType        <string>
      fieldsV1  <FieldsV1>
      manager   <string>
      operation <string>
      subresource       <string>
      time      <string>
    name        <string>
```

Pods

```
ubuntu@arm:~$ kubectl explain pod.spec.affinity
KIND:       Pod
VERSION:    v1

FIELD: affinity <Affinity>


DESCRIPTION:
    If specified, the pod's scheduling constraints
    Affinity is a group of affinity scheduling rules.

FIELDS:
  nodeAffinity  <NodeAffinity>
    Describes node affinity scheduling rules for the pod.

  podAffinity   <PodAffinity>
    Describes pod affinity scheduling rules (e.g. co-locate this pod in the same
    node, zone, etc. as some other pod(s)).

  podAntiAffinity       <PodAntiAffinity>
    Describes pod anti-affinity scheduling rules (e.g. avoid putting this pod in
    the same node, zone, etc. as some other pod(s)).
```

Pods – Labels e Anotações

- Labels permitem você associar os Pods à outros objetos (Serviços, Ingress, Controladores, etc)

- Anotações permitem você adicionar características experimentais ou integrar com ferramentas terceiras

```
apiVersion: v1
kind: Pod
metadata:
  name: aula-pods
  labels:
    regiao: sul
    versao: beta
spec:
  containers:
    - name: ubuntu
      image: ubuntu:22.04
      command: ['tail', '-f', '/dev/null']
  restartPolicy: OnFailure
```

Você pode utilizar essas labels para afinidade ou anti-afinidade.

Também podemos associar a serviços

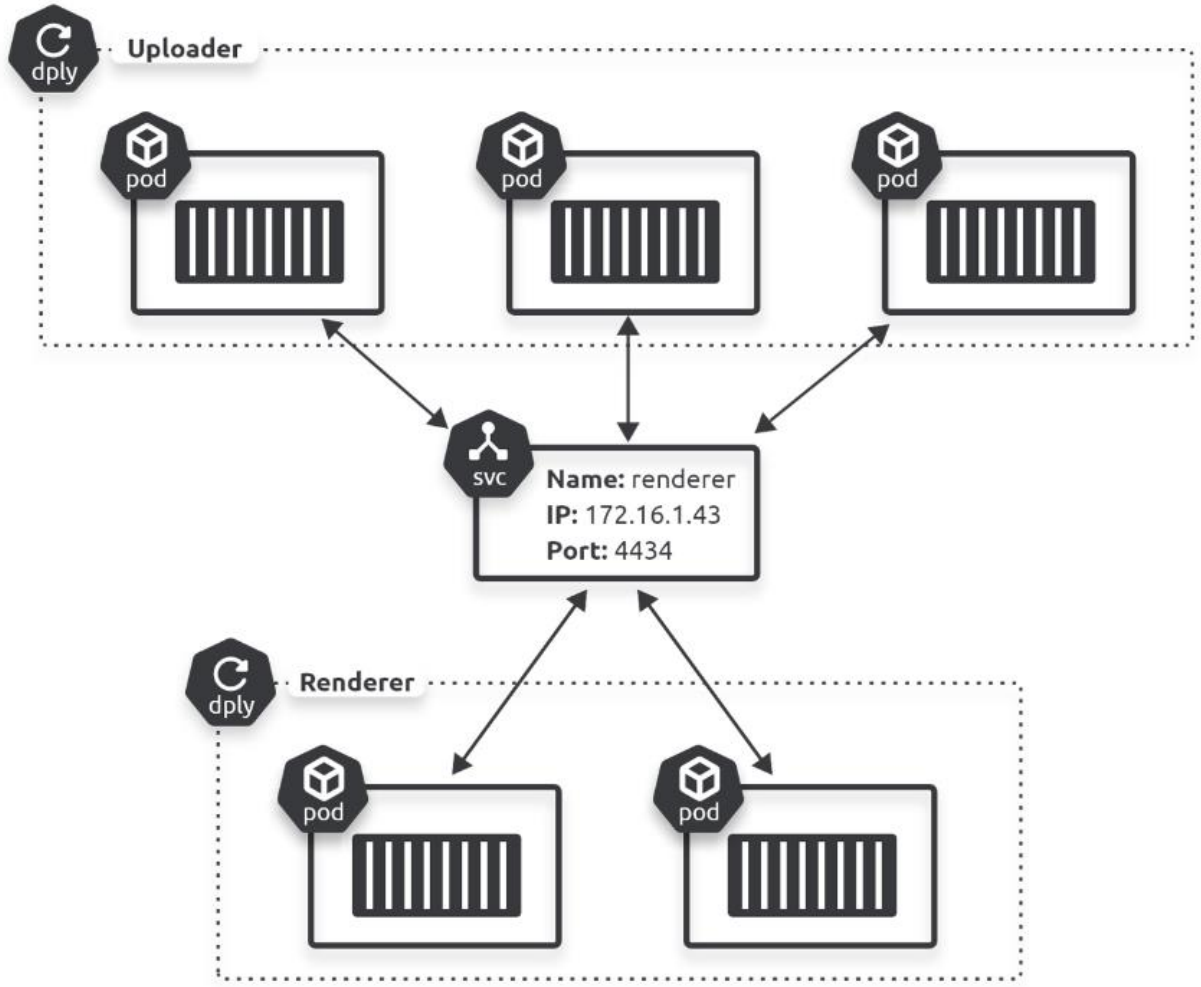
Serviços

Serviços

- Com os Pods entendemos que eles possuem um nome e endereço IP únicos
  - Também entendemos que eles são efêmeros
- Pods são substituídos a cada rollout, dimensionamento ou falhas
  - Isso torna seu acesso instável
- Exemplo
  - Você possui um microsserviço com uma série de Pods renderizando um vídeo
  - Como isso funcionaria caso não pudesse confiar em todas as partes quando solicitadas?

Serviços

- Resolvemos o cenário anterior com um objeto denominado Service (Serviço)
  - Fornecem uma interface estável para trabalhar com redes sobre um conjunto de Pods
- Ele também é capaz de balancear cargas entre diversos Pods



Fonte: POULTON, Nigel; JOGLEKAR, Pushkar. **The Kubernetes book**. Version 4, March 2019. United Kingdom: Nigel Poulton, 2019.

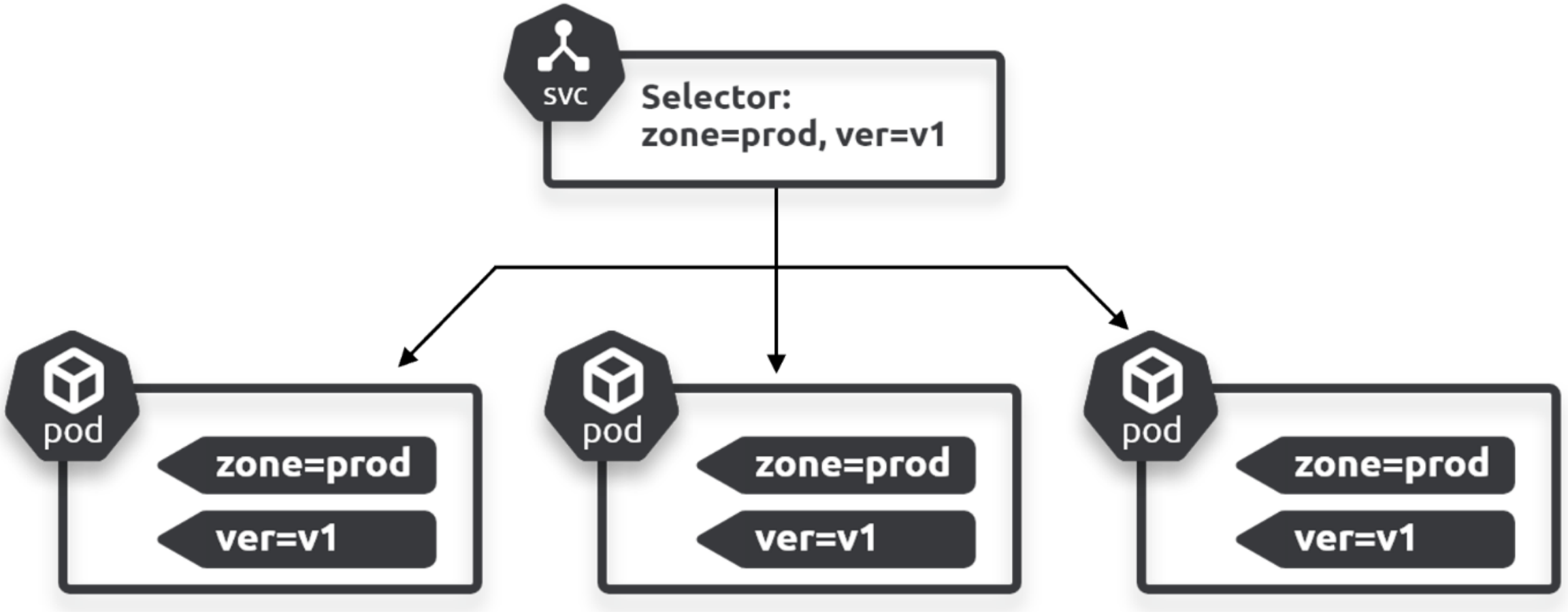
Serviços

- Na visão “front-end” do serviço enxergamos somente um nome DNS, endereço IP e porta

- Na visão “back-end” acontecem mais coisas
  - Balanceamento de carga entre um conjunto dinâmico de Pods
  - Pods são adicionados/removidos ao serviço transparentemente
  - Fornecem roteamento a nível de TCP/UDP
    - Não possuem inteligência de aplicação!
    - Não são capazes de rotear a nível de aplicação (Podemos utilizar um Ingress)

- A associação entre Pods e Services é realizada através de labels
  - Acoplamento fraco

Serviços – Exemplo de associação com labels

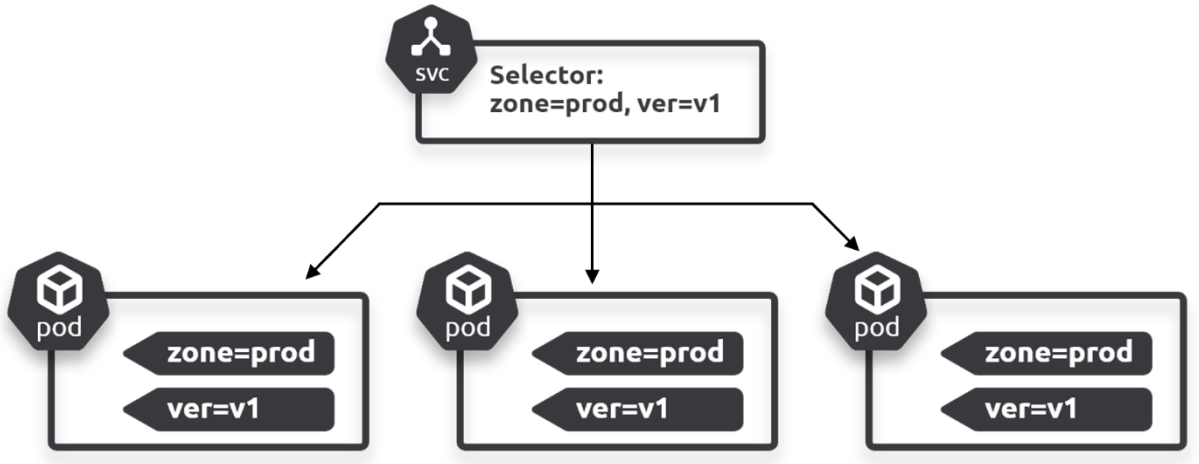


Fonte: POULTON, Nigel; JOGLEKAR, Pushkar. **The Kubernetes book**. Version 4, March 2019. United Kingdom: Nigel Poulton, 2019.

Serviços – Exemplo de associação com labels

- Aqui a associação é realiada utilizando o operador E (lógica booleana)

- "Se associe aos Pods com `zone=prod` **E** `ver=v1`"

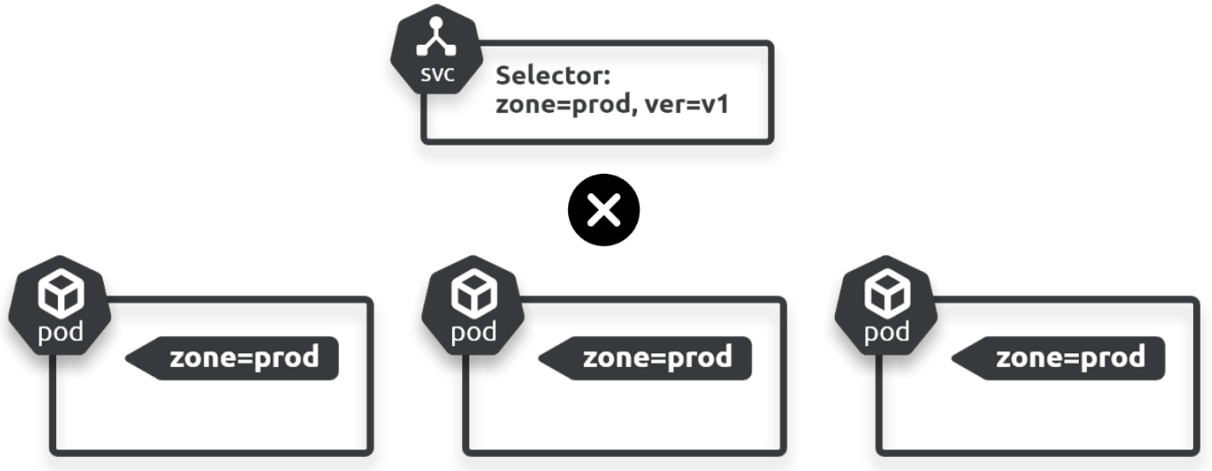


Fonte: POULTON, Nigel; JOGLEKAR, Pushkar. **The Kubernetes book**. Version 4, March 2019. United Kingdom: Nigel Poulton, 2019.

Serviços – Exemplo de associação com labels

- Exemplo sem correspondência

- Procura por `zone=prod` E `ver=1`
  - Encontrou somente `zone=prod`



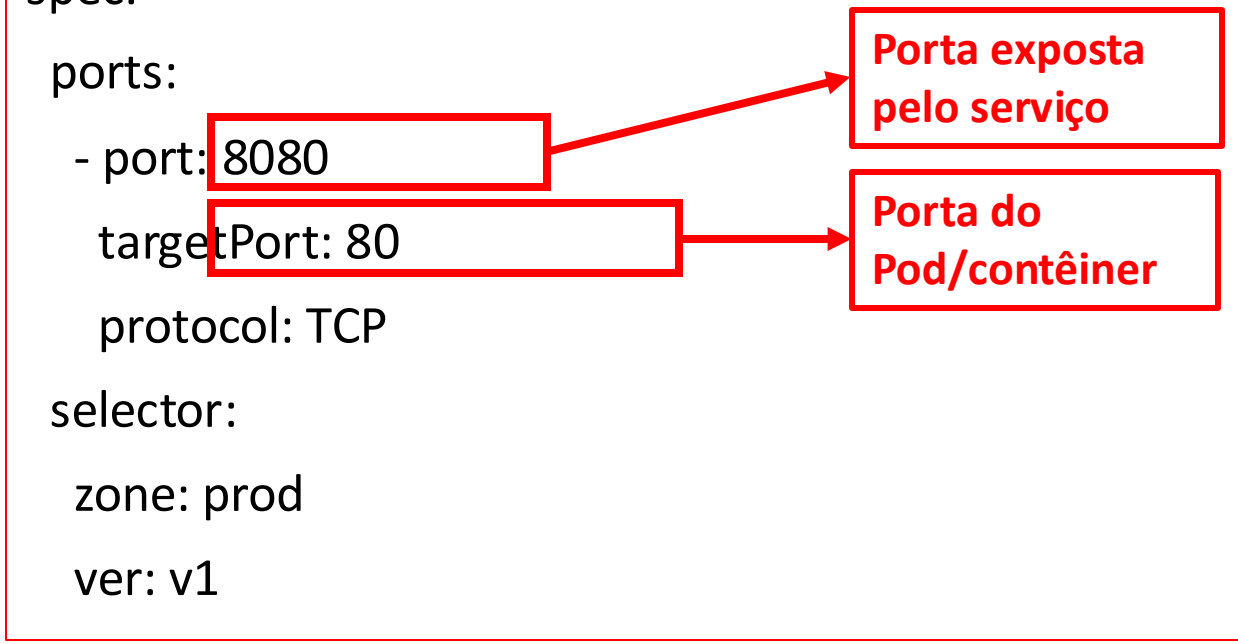
Fonte: POULTON, Nigel; JOGLEKAR, Pushkar. **The Kubernetes book**. Version 4, March 2019. United Kingdom: Nigel Poulton, 2019.

Serviços – Exemplo de associação com labels

- Manifesto do Service utilizado anteriormente

- Mas como associamos aos Pods?
  - Criamos essas labels nos metadados!

```
apiVersion: v1
kind: Service
metadata:
  name: aula-servico
spec:
  ports:
   - port: 8080
     targetPort: 80
     protocol: TCP
  selector:
    zone: prod
    ver: v1
```



Serviços

- Vamos testar um exemplo

```
wget -qO svc-discovery.yml https://shorturl.at/lnrQE
```

- Aplicar o comando no Kubernetes

```
kubectl apply -f svc-discovery.yml
```

- Acessar um dos Pods

```
kubectl exec -it -n dev dev -- bash
```

- Instalar cURL e acessar um serviço

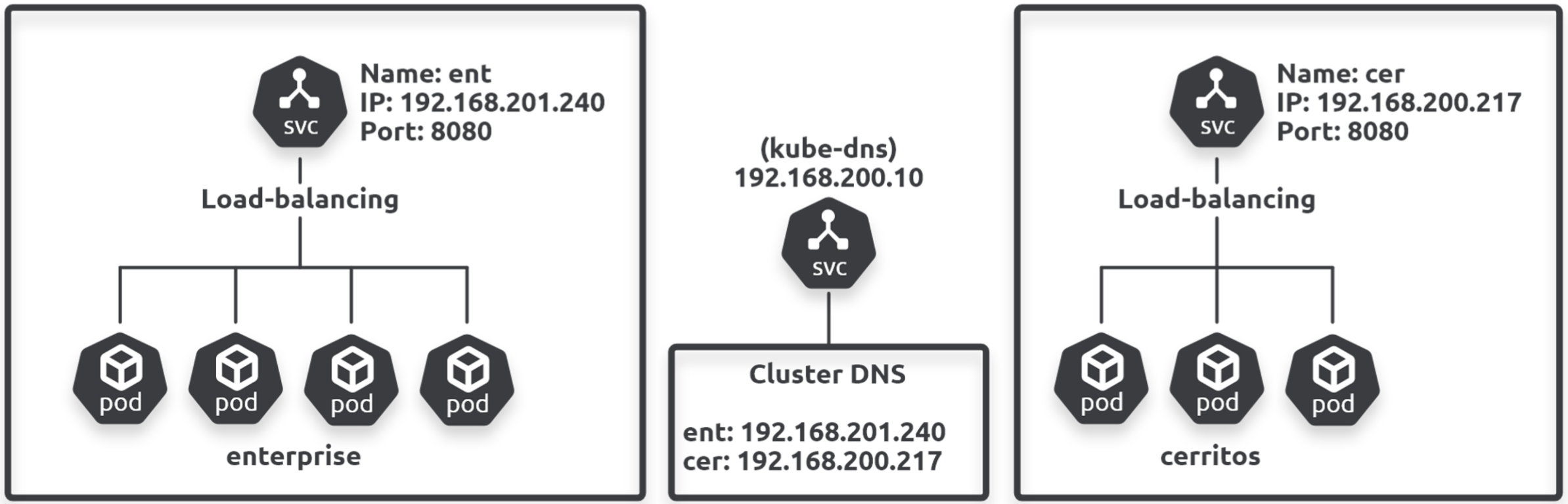
```
apt update && apt install curl -y
curl dev.dev.svc.cluster.local
curl prod.prod.svc.cluster.local
```

Serviços

- Quando trabalhamos com microsserviços queremos conectá-los uns aos outros

- O Kubernetes possui um mecanismo de descoberta de serviços baseado por DNS entre Pods e Serviços
  - Kube-dns (frontend do DNS do cluster, implementa lógica a nível de cluster)
  - Coredns (backend do DNS do cluster, imeplementa lógica a nível de nó)

Serviços

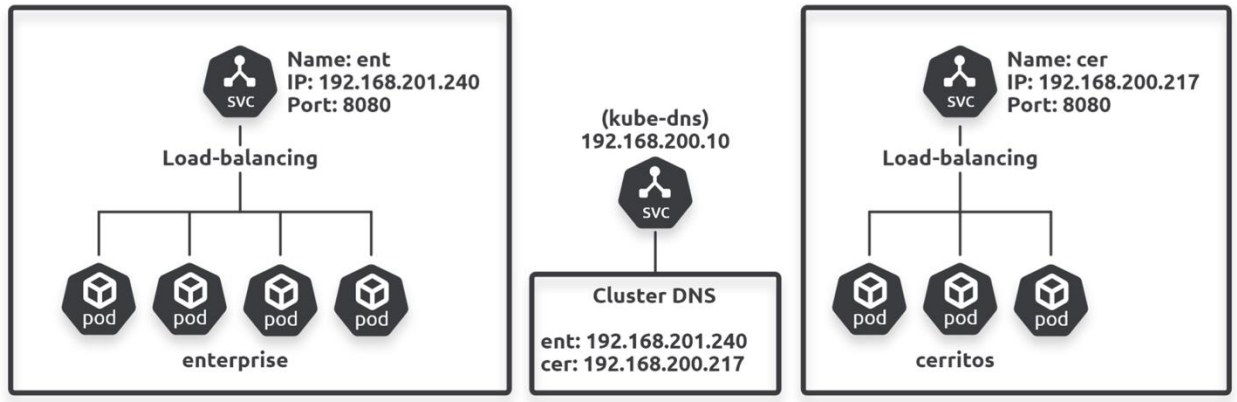


Fonte: POULTON, Nigel; JOGLEKAR, Pushkar. **The Kubernetes book**. Version 4, March 2019. United Kingdom: Nigel Poulton, 2019.

Serviços

- Nós conseguimos identificar os Pods através do IP dos serviços, mas há um problema:
  - Como saber o IP do serviço?
  - Como tratar situações onde um Pod depende de outro?

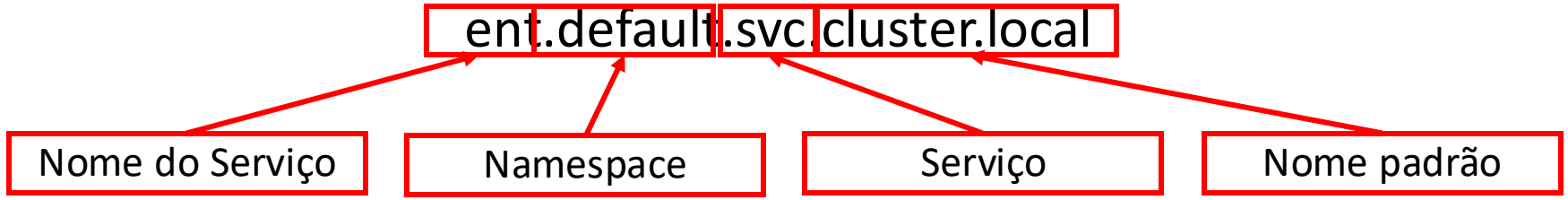
- Aqui temos a noção de Nome de Domínio Completamente Qualificado (FQDN)
  - Não precisamos especificar o IP do Pod!

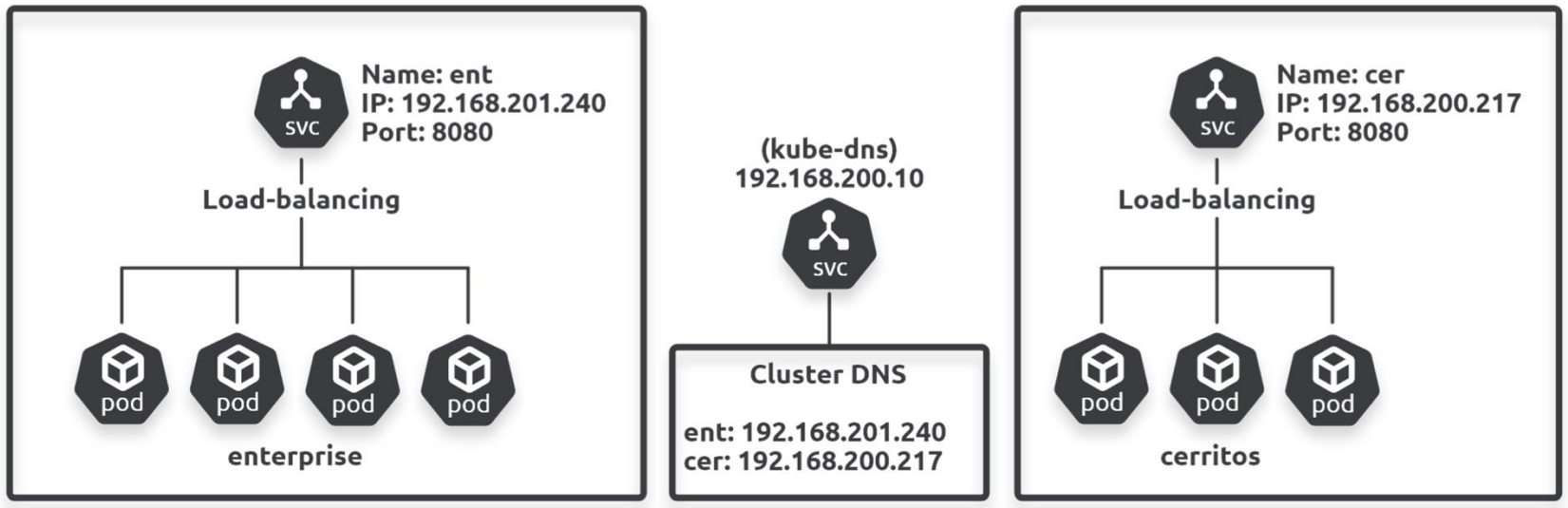


Fonte: POULTON, Nigel; JOGLEKAR, Pushkar. **The Kubernetes book**. Version 4, March 2019. United Kingdom: Nigel Poulton, 2019.

Serviços

- Nesse esquema podemos especificar um domínio seguindo a seguinte sintaxe:





Fonte: POULTON, Nigel; JOGLEKAR, Pushkar. **The Kubernetes book**. Version 4, March 2019. United Kingdom: Nigel Poulton, 2019.

Serviços

- Temos alguns tipos de Services
  - NodePort
  - ClusterIP
  - LoadBalancer
  - ExternalName

- Os tipos dos serviços especificam como será seu funcionamento

Serviços

- De acordo com a documentação do Kubernetes:
  - ClusterIP
    - “Expõe um IP interno do cluster e é alcançável somente do cluster. Configuração padrão de serviços”
  - NodePort
    - “Expõe uma porta em cada IP do nó a uma porta estática (NodePort).”
  - LoadBalancer
    - “Expõe um serviço externamente através de um balanceador de carga externo. O Kubernetes não fornece um componente de balanceamento”
  - ExternalName
    - “Mapeia o serviço ao conteúdo do campo externalName. Configura o DNS do Kubernetes”

Serviços – ClusterIP

ClusterIP

- "Expõe um IP interno do cluster e é alcançável somente do cluster. Configuração padrão de serviços"

Aqui conseguimos acessar somente de dentro do cluster

Definimos o campo `clusterIP` como `None` se

- Não precisamos/queremos usar load-balancing do Kubernetes
- Mais de um IP por serviço

```
apiVersion: v1
kind: Service
metadata:
  name: teste-clusterip
spec:
  ports:
    - port: 80
      targetPort: 80
  selector:
    app: teste-clusterip
```

Serviços – NodePort

NodePort

- "Expõe uma porta em cada IP do nó a uma porta estática (NodePort)."

As portas devem ser entre 30000 e 32767

O serviço será exposto a porta 30000 associada ao IP do nó que o Pod está alocado

```
apiVersion: v1
kind: Service
metadata:
  name: teste-nodeport
spec:
  type: NodePort
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30000
  selector:
    app: teste-nodeport
```

Serviços – LoadBalancer

LoadBalancer

- "Expõe um serviço externamente através de um balanceador de carga externo. O Kubernetes não fornece um componente de balanceamento"

Nesse tipo de serviço é necessário configurar um *Load Balancer* que aponte ao cluster

```
apiVersion: v1
kind: Service
metadata:
  name: aula-servico
spec:
  type: LoadBalancer
  ports:
    - port: 8080
      nodePort: 5000
  selector:
    app: teste-loadbalancer
```

Serviços – ExternalName

ExternalName

- "Mapeia o serviço ao conteúdo do campo externalName. Configura o DNS do Kubernetes"
- Aqui podemos fazer um mapeamento para um CNAME especificado
- Aplicável a nível de namespace

```
apiVersion: v1
kind: Service
metadata:
  name: prod
  namespace: prod
spec:
  type: ExternalName
  externalName: google.com
```

Atividade

Atividade

- Utilizar Pod criado anteriormente e associá-lo a um serviço

- Torná-lo acessível pelo navegador utilizando um serviço do tipo NodePort

Referências

Referências

1. https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands
2. https://microk8s.io/docs/getting-started
3. https://kubernetes.io/docs/concepts/services-networking/service/
4. https://kubernetes.io/pt-br/docs/concepts/overview/components/
5. https://kubernetes.io/pt-br/docs/concepts/overview/working-with-objects/
6. https://kubernetes.io/pt-br/docs/concepts/overview/

Referências

7. POULTON, Nigel; JOGLEKAR, Pushkar. **The Kubernetes book**. Version 4, March 2019. United Kingdom: Nigel Poulton, 2019.
8. RICE, Liz. **Container security: fundamental technology concepts that protect containerized applications**. First edition. Beijing Boston Farnham Sebastopol Tokyo: O'Reilly, 2020.

Contato: p.horchulhack@pucpr.br